Escuela Politécnica Superior

Creatividad computacional: Diseño y desarrollo de un método para generación automática de descripciones asociadas a elementos de juegos de rol

Grado en Ingeniería Informática

Trabajo Fin de Grado

Autor:
Alejandro Vicente Rodríguez Segado

Tutor/es:
Elena Lloret Pastor

Universitat d'Alacant
Universidad de Alicante

Julio 2021

Creatividad computacional

Diseño y desarrollo de un método para generación automática de descripciones asociadas a elementos de juegos de rol

Autor
Alejandro Vicente Rodríguez Segado

Directores
Elena Lloret Pastor
Lenguajes y Sistemas Informáticos

Grado en Ingeniería Informática

Escuela
Politécnica
Superior

ALICANTE, 30 de julio de 2021

Resumen

La capacidad creativa de las máquinas es un asunto que siempre ha despertado controversia, tanto en medios divulgativos, como en expertos en la materia. Hoy en día, pese a la proliferación de aplicaciones creativas en el campo de la creatividad computacional, las opiniones continúan estando divididas, algunas de ellas poniendo en duda el potencial creativo de las máquinas y tachándolas de "expertas falsificadoras". Pero acaso, ¿no somos las personas también "expertos falsificadores"?

Este trabajo, pretende experimentar con la capacidad creativa de las máquinas para la generación de lenguaje natural (GLN). Este campo pertenece a la disciplina conocida como lingüística computacional (LC) o procesamiento del lenguaje natural (PLN), cuyo propósito es entender y generar lenguaje natural. Así, con el fin de poner a prueba como se desenvuelven las máquinas con esta habilidad inherente en los humanos, se pretende diseñar un sistema capaz de generar automáticamente descripciones asociadas a juegos de rol.

Tras una exhaustiva investigación del estado actual de la tarea de la GLN, el enfoque propuesto para abordar el sistema descrito se fundamenta en las tecnologías de vanguardia basadas en modelos de lenguaje neuronales (*NLM: Neural Language Models*), concretamente en el modelo GPT-2 de la arquitectura Transformer. Estas tecnologías se profundizan en el capítulo 3 de este trabajo.

En el capítulo 7 se presenta una evaluación de los textos generados por el sistema, utilizando distintas técnicas de decodificación. A pesar de estar entrenado con una escasa cantidad de datos, el sistema es capaz de generar algunos textos creativos y gramaticalmente correctos. Como se precisa también en el capítulo 3, la cantidad de datos requeridos para entrenar un modelo de lenguaje desde cero, como en este caso, es extremadamente grande, por lo que los investigadores suelen utilizar modelos preentrenados. Puesto que la mayoria de los recursos para abordar la tarea de la GLN, así como otros campos de la LC, están en inglés, y el sistema propuesto es sumamente específico, la alternativa de recurrir a un modelo preentrenado se dificulta. Por estos motivos, el enfoque propuesto para este sistema es un entrenamiento desde cero (*from scratch*).

Palabras clave: creatividad, creatividad computacional, lenguaje natural, procesamiento de lenguaje natural, lingüística computacional, generación de lenguaje natural, GPT-2, modelos de lenguaje, language models

Agradecimientos

A mi tutora Elena Lloret Pastor,
por su tiempo, dedicación y consejos,
y sobretodo, por su comprensión,
durante todo este tiempo.

A mi compañera en la vida,
por estar siempre a mi lado, apoyándome,
en los buenos y malos momentos,
y confiar siempre en mí.

A mi familia,
por haberme soportado durante largos años,
o al menos, haberlo intentado.

Vida antes que muerte.
Fuerza antes que debilidad.
Viaje antes que destino.

Brandon Sanderson, El camino de los reyes.

Índice general

Índice de figuras

Índice de tablas

1 Introducción

La creatividad y el lenguaje natural son dos ideas que no se asocian naturalmente con los ordenadores. Sin embargo, en los últimos años se ha producido una oleada de investigaciones sobre la combinación de ambos temas. Este proyecto trata de indagar en como achacar ambas materias al ámbito computacional, con el fin de poner a prueba el potencial creativo de las máquinas. Para ello, se pretende diseñar un sistema que genere automáticamente descripciones asociadas a juegos de rol en español.

Para entrar en contexto, en este capítulo se precisa el concepto de creatividad y se presenta una recapitulación de cómo ha ido evolucionando esta disciplina en el ámbito computacional a lo largo de la historia.

Según Boden (1998):

> "La creatividad no es una "facultad" especial, ni una propiedad psicológica limitada a una pequeña élite. Más bien es una característica de la inteligencia humana en general. Se basa en capacidades cotidianas como la asociación de ideas, el recuerdo, la percepción, el pensamiento analógico, la búsqueda de un espacio problemático estructurado y la autocrítica reflexiva. No sólo implica una dimensión cognitiva (la generación de nuevas ideas), sino también la motivación y la emoción, y está estrechamente relacionada con el contexto cultural y los factores de personaldiad"

Como sostiene Colton et al. (2009):

> "En esencia, la creatividad computacional es el estudio de la creación de software que exhibe un comportamiento que se consideraría creativo en humanos. Este software creativo se puede utilizar para áreas creativas autónomas, como inventar teorías matemáticas[1], escribir poemas[2], pintar cuadros[3] y componer música[4]. Sin embargo, los estudios de creatividad computacional también nos permiten comprender la creatividad humana y producir programas para que los utilicen personas creativas, donde el software actúa como un colaborador creativo en lugar de una mera herramienta".

Tradicionalmente, ha sido difícil para la gente aceptar máquinas que pretender ser inteligentes y aún más difícil admitir que pueden ser creativas. Por ejemplo, en 1934, algunos profesores de la Universidad de Manchester en el Reino Unido construyeron

[1] https://www.nature.com/articles/d41586-021-00304-8
[2] https://www.lavanguardia.com/tecnologia/20190504/462008315059/inteligencia-artificial-robot-misica-arquitectura.html
[3] https://es.gizmodo.com/una-inteligencia-artificial-pinta-un-nuevo-cuadro-de-re-1769869684
[4] https://www.elmundo.es/tecnologia/2016/09/26/57e93953e5fdea42278b4658.html

modelos meccano que podían resolver algunas ecuaciones matemáticas[5]. Pionero para su época, este proyecto fue escrito en un artículo de Meccano Magazine[6]. Y era muy optimista, pero sorprendentemente concluía con la siguiente afirmación: "*El pensamiento verdaderamente creativo, por supuesto, siempre permanecerá más allá de cualquier máquina*". Sin embargo, con los avances tecnológicos de las últimas décadas, la gente, incluso dentro de la informática, sigue escéptica con la creatividad potencial del software. Por ejemplo, en Non-Photorealistic Rendering[7]. Un libro de texto de gráficos publicado en el año 2000, los autores Thomas Strothotte y Stefan Schlechtweg afirman audazmente que: "*Simular ténicas artísticas significa también simular el pensamiento y razonamiento humano, especialmente pensamiento creativo. Esto es imposible de hacer utilizando algoritmos o sistemas de procesamiento de información*". No obstante, esta última década la creatividad computaconal ha proliferado a un ritmo vertiginoso alcanzando su madurez como disciplina y aumentando la controversia que despertaban las afirmaciones que ponían en entredicho la creatividad potencial de las máquinas. Esta madurez es evidente debido a la gran cantidad de actividad dentro del campo de la inteligencia artificial durante los últimos años; la calidad y variedad del software producido; el valor cultural de los artefactos producidos por dicho software; y lo más importante, el consenso que está estableciéndose sobre cuestiones generales (Colton et al., 2009). Hoy en día, existen sistemas capaces de crear imagenes fotorealistas a partir de un garabato[8]. También los hay que escriben poesía a partir de imágenes o que escriben libros[9].

[5] https://felixmaocho.wordpress.com/2009/07/06/mi-viejo-meccano-ordenador-analogico-de-meccano/
[6] https://en.wikipedia.org/wiki/Meccano_Magazine
[7] https://www.amazon.es/gp/product/B00CXO9700/ref=dbs_a_def_rwt_bibl_vppi_i0
[8] https://www.xataka.com/robotica-e-ia/esta-ia-solo-necesita-garabato-cara-para-crear-retrato-fotorrealista
[9] https://www.yorokobu.es/creatividad-computacional/

2 Objetivos

2.1. Generales

Todo trabajo tiene com objetivo la adquisición y difusión de conocimiento. En el ámbito de este proyecto, se desea aprender las nociones básicas sobre la lingüística computacional (LC). En esencia su aplicación a la generación de lenguaje natural (GLN). Además, se pretende crear un sistema que genere descripciones automáticas asociadas a elementos de juegos de rol.

2.2. Específicos

Además del objetivo principal comentado anteriormente, este TFG tiene los siguientes objetivos específicos:

- Analizar sistemas y aplicaciones de GLN existentes.
- Aprender cuáles son las técnicas de Inteligencia Artificial más adecuadas para abordar el TFG.
- Analizar y decidir las tecnologías más apropiadas para el desarrollo del trabajo.
- Aprender y utilizar herramientas y liberías específicas de Inteligencia Artificial y Procesamiento de Lenguaje Natural, así como frameworks para diseñar arquitecturas (por ejemplo, TensorFlow, NTLK).
- Analizar y desarrollar un método que permita generar lenguaje natural.
- Evaluar el método propuesto para determinar la calidad y adecuación del texto generado.

3 Marco Teórico

Como se dijo anteriormente, la motivación tras este trabajo es diseñar un sistema que genere automáticamente descripciones asociadas a juegos de rol en español, con el fin de poner a prueba la creatividad (lingüística) de las máquinas. El enfoque propuesto para abordar este sistema se basa en el estado actual del arte. El modelo GPT-2 de la arquitectura transformer. Asimismo, se expuso una recapitulación de la evolución de la creatividad computacional a lo largo de los años.

En este capitulo se realiza una revisión sobre el trabajo previo a este proyecto. En primera instancia se definen los conceptos detrás del campo de la GLN, con la pretensión de esclarecer los fundamentos de esta disciplina (sección 3.1). Tras esta primera toma de contacto, en la sección 3.3 se realiza un análisis de los enfoques tradicionales para abordar la tarea de la GLN (enfoques basados en reglas o plantillas y enfoques estadísticos) y del paradigma actual, los modelos de lenguaje neuronales (*NLM: Neural Language Models*). Además, en esta sección se presenta el concepto de modelo de lenguaje (*LM: Language Model*) y las redes neuronales más relevantes para la GLN como las redes neuronales recurrentes (RNN), LSTM y la innovadora arquitectura Transformer. Esta última se exhibe en una sección independiente (3.5) a causa de su relevancia para este trabajo. Para finalizar, se exponen algunas aplicaciones dedicadas a la tarea de la GLN similares al sistema descrito.

3.1. Generalidades

3.1.1. Definición de lenguaje

Un lenguaje se puede definir de diferentes formas: desde el punto de vista funcional lingüístico se define como una función que expresa pensamientos y comunicaciones entre la gente. Esta función puede realizarse mediante signos escritos (escritura) o mediante señales y vocales (voz). Desde un punto de vista formal se define como un conjunto de frases, que generalmente es infinito y se forma con combinaciones de elementos tomados de un conjunto (usualmente infinito) llamado alfabeto, respetando un conjunto de reglas de formación (sintáctica o gramaticales) y de sentido (semánticas). Además de las características fundamentales del lenguaje debe considerarse que sea funcional, es decir, el lenguaje debe permitirnos expresar nuestras ideas. El lenguaje será bueno en la medida que sea fácil de leer, fácil de entender y fácil de modificar. lo mismo ocurre en los lenguajes formales (Cortez Vásquez et al., 2009).

Podemos distinguir entre dos clases de lenguajes: los lenguajes naturales (inglés, alemán, español, etc.) y lenguajes formales (matemático, lógico, progamable, etc.).

3.1.2. Definición de lenguaje natural

El lenguaje natural (LN) es el medio que utilizamos de manera cotidiana para establecer nuestra comunicación con las demás personas. El LN ha venido perfeccionándose a partir de la experiencia a tal punto que puede ser utilizado para analizar situaciones altamente complejas y razonar muy sutilmente. Los lenguajes naturales tienen un gran poder expresivo y su función y valor como una herramienta para razonamiento. Por otro lado, la sintaxis de un LN puede ser modelada fácilmente por un lenguaje formal, similar a los utilizados en las matemáticas y la lógica (Cortez Vásquez et al., 2009).

3.1.3. Definición de lenguaje formal

El lenguaje formal es aquel que el hombre ha desarrollado para expresar las situaciones que se dan en específico en cada área del conocimiento científico. Los lenguajes formales pueden ser utilizados para modelar una teoría de la mecánica, física, matemática, ingeniería eléctrica, o de otra naturaleza, con la ventaja de que en estos toda ambigüedad es eliminada (Cortez Vásquez et al., 2009).

3.1.4. Lingüística Computacional

La Lingüística Computacional (LC) es un campo en el que convergen diversas disciplinas: la lingüística aplicada, la informática y la inteligencia artificial. Se ocupa de las interacciones entre los ordenadores y los lenguajes humanos, por lo que en ocasiones se refiere también como Procesamiento de Lenguaje Natural (PLN), abarcando la comprensión del lenguaje y su generación.

Siendo un área de investigación en continuo desarrollo, la LC se ocupa en la actualidad de un conjunto de tareas cuyo propósito es comprender y/o producir lenguaje natural. Lo que busca, entonces, es conseguir sistermas, capaces tanto de entender el significado del lenguaje humano (lenguaje natural) como de generarlo. Entre las citadas tareas cabe destacar la traducción automática, los sistemas de recuperación de información, la elaboración automática de resúmenes, o la generación de lenguaje natural (Vicente et al., 2015).

3.2. Generación de Lenguaje Natural

La Generación de Lenguaje Natural (GLN) es una tarea multidisciplinar. Su investigación y desarrollo incorpora conocimiento procedente de áreas diversas como la lingüística, la psicología, la ingeniería y la informática. El objetivo principal de la disciplina es investigar cómo se pueden crear aplicaciones informáticas capaces de producir por si mismas textos de alta calidad en lenguaje natural. Para ello parte o bien de representaciones de datos estructurados y procesables (ficheros binarios, datos numéricos, bases de datos, etc.) o bien de textos escritos en lenguaje natural. La transformación de estos datos no puede ser realizada de manera directa, se han de tomar muchas decisiones relativas a diferentes aspectos como la determinación del contenido del mensaje y su estructura, la

relaciones retóricas en varios niveles (texto, párrafo, frase), la elección de las palabras adecuadas, la disposición final del texto (títulos, cabeceras, pies de página, etc.) o los patrones acústicos en el caso de que la salida final del mensaje sea oral. Uno de los mayores desafíos de la GLN es la construcción de arquitecturas en las que se puedan tomar todas estas decisiones de manera que sea temporalmente abordable la producción de textos, ya sea en formato de texto o audio. Respecto a las aplicaciones de la GLN, estás son muy amplicas y variadas. Existen en el mercado sistemas que se encargan por ejemplo de la generación de partes metereológicos. También los hay que generan manuales de instrucciones, informes de evaluaciones académicas o crean resúmenes (Vicente et al., 2015).

3.3. Evolución de la GLN

Después de aclarar los conceptos detrás del ámbito de este trabajo, es fundamental entender la evolución de los métodos para abordar la tarea de la GLN a lo largo de los años. En este apartado se presenta una recapitulación de los métodos más destacados para desempeñar la tarea de la GLN: iniciando con los enfoques más clásicos, como los enfoques basados en reglas o plantillas y los enfoques estadísticos; y concluyendo con el estado actual del arte, los modelos de lenguaje neuronales, conocidos como *NLM (Neural Language Models)*. Cabe mencionar que el proceso evolutivo que se pretende relatar no atañe únicamente al campo de la GLN, sino que concierne a otras aplicaciones de la LC: respuesta a preguntas, resumenes de textos, identificación de entidades, clasificación de documentos, desambigüación de términos, etc (Vicente et al., 2015) (Vaca, 2021).

Como se indicó anteriormente, los enfoques tradicionales para emprender la tarea de la GLN son los enfoques basados en reglas o plantillas y los enfoques estadísticos. Vicente et al. (2015) presenta una descripción muy esclarecedora de uno y otro. Sin embargo denomina a los enfoques basados en reglas o plantillas, como enfoques basados en el conocimiento:

> *"Por un lado, hablamos de sistemas basados en conocimiento cuando las técnicas que lo implementan se nutren de fuentes con un marcado carácter lingüístico como diccionarios, tesauros, bases de conocimiento léxicas, reglas o plantillas. De esos recursos se extrae la información morfológica, léxica, sintáctica, semántica, etc. Por otro lado, cuando un sistema se desarrolla bajo un enfoque estadístico, la información que necesita para transformar la entrada a un texto en lenguaje natural procede principalmente de un corpus y de las probabilidades extraídas de los textos que lo componen, pudiendo estos estar etiquetados o no. Estos sistemas están menos restringidos a un dominio o un idioma que los basados en conocimiento, dado que, si el corpus empleado es adecuado tanto en tamaño como en tipo de contenido, no tiene tantas restricciones como las que resultan de generar unas reglas que han de ceñirse a las características del contexto para el que se desarrollan o a las peculiaridades de una lengua concreta, como suele ser el caso de los sistemas basados en conocimiento."*

A mitad de la década del 2010, con la proliferación del *Deep Learning*[10] y la irrupción de los *Word Embeddings*[11], los enfoques clásicos para abordar la tarea de la GLN cedieron el paso a los modelos lingüísticos neuronales. Así, los modelos basados en redes neuronales recurrentes (RNN) que hacian uso de *Word Embeddings* se convirtieron en el estándar, al menos hasta finales del año 2017, pues eran capaces no sólo de tener en cuenta el significado general de la palabra, sino también de la posición de la misma en la frase.

A finales de 2017, en un *papper* de Google conocido como *Attention Is All You Need*[12] se presentó la arquitectura Transformer. Un modelo que tenía como principal innovación la sustitución de las capas recurrentes como las LSTMs que se venían usando hasta el momento en la LC, por las denominadas capas de atención[13].

Estas capas de atención codifican cada palabra de una frase en función del resto de la secuencia, permitiendo así introducir el contexto en la representación matemática del texto, motivo por el cual los modelos basados en la arquitectura Transformer se les denomina Embedding Contextuales[14].

Desde su irrupción en el paradigma de la LC, los modelos basados en la arquitectura Transformer se han convertido en el nuevo estado del arte en la tarea de análisis de texto, siendo así la familia de modelos que mejores resultados han obtenido en diferentes campos de la LC, como la GLN.

En las secciones posteriores, se presentan los conceptos más relevantes de la recapitulación previa.

3.4. LM: Language Models

Un *LM* es básicamente una distribución de probabilidad sobre palabras o secuencias de palabras. En la práctica, un modelo lingüístico da la probabilidad de que una determinada secuencia de palabras sea válida. En este contexto, la validez no se refiere en absoluto a la validez gramatical. Significa que se asemeja a la forma en que la gente habla (o, para ser más precisos, escribe) que es lo que aprende el modelo lingüístico (Kapronczay, 2021).

Existen principalmente dos tipos de modelos lingüísticos (Kapronczay, 2021):

1. **Modelos de lenguaje estadísticos:** Estos modelos utilizan técnicas estadísticas tradicionales como los n-gramas (un n-grama es una subsecuencia de n delementos de una secuencia dada), los modelos ocultos de Markov[15] (HMM: Hidden Markov Models) y ciertas reglas lingüísticas para aprender la distribución de las palabras. Cabe destacar que los *LM* son una herramienta fundamental de los enfoques es-

[10] https://es.wikipedia.org/wiki/Aprendizaje$_p rofundo$
[11] https://es.wikipedia.org/wiki/Word$_e mbedding$
[12] https://arxiv.org/pdf/1706.03762.pdf
[13] https://www.wolfram.com/language/12/neural-network-framework/use-transformer-neural-nets.html.es
[14] https://stackoverflow.com/questions/62272056/what-are-the-differences-between-contextual-embedding-and-word-embedding
[15] https://www.redalyc.org/pdf/993/99324907003.pdf

tadísticos ya mencionados, por lo que puede referirse a ellos como modelos de lenguaje estadísticos. En el apartado 3.4.2 se cita algún ejemplo de estos sistemas.

2. **Modelos de lenguaje neuronales:** Anteriormente dicho, los NLM son la tecnología de vanguardia del paradigma de la GLN y han superado la eficacia de los *LM* estadísticos. Utilizan diferentes tipos de redes neuronales para modelar el lenguaje. En 3.4.3 se precisan las arquitecturas de redes neuronales más relevantes para abordar la tarea de la GLN como la RNN, LSTM, y la vanguardista arquitectura Transformer, y se cita algún ejemplo de cada una de ellas.

3.4.1. Enfoques basados en reglas o plantillas

En esta sección se presentan los sistemas WMO-based y NATURAL (Gkatzia et al., 2017). Estos sistemas basados en reglas están dedicados a la tarea de GLN. Concretamente, generan descripciones textuales de los datos de precipitaciones y temperatura que abordan la naturaleza incierta de las previsiones.

Por un lado, WMO-based es un sistema basado en reglas que utiliza las directrices de la OMM[16] para informar de la incertidumbre, como muestra la tabla 3.1. Este sistema mapea directamente las probabilidades del tiempo a la correspondiente interpretación lingüística. Por ejemplo, considerando una previsión de intervalos soleados con un 30 % de probabilidad de lluvia, este sistema generará la siguiente previsión: “Intervalos soleados con lluvia possible - menos probable que no”.

Sin embargo, NATURAL, pese a compartir el mismo enfoque que el sistema anterior, trata de imitar a los pronosticadores en su forma habitual de informar sobre el tiempo. Las reglas utilizadas en este sistema se han derivado de la observación de la forma en que los expertos (por ejemplos, reporteros de la BBC[17]) producen pronósticos. Para el ejemplo anterior (intervalos soleados con un 30 % de probabilidad de lluvia), este sistema generara la siguiente previsión: ”Principalmente secon con intervalos soleados”. Este sistema es más natural que el basado en OMM en el sentido de que las probabilidades se asignan a la interpretación lingüística del tiempo (por ejemplo, ïntervalos soleados”) en lugar de la asignación lingüística de la incertidumbre (por ejemplo, ”probable”).

[16] https://public.wmo.int/es
[17] https://www.bbc.com/mundo

Probabilidad de ocurrencia	Lexicalización
$p > 0.99$	“extremadamente probable”
$0.90 \leq p \leq 0.99$	“muy probable”
$0.70 \leq p \leq 0.89$	“probable”
$0.55 \leq p \leq 0.69$	“probable - más probable que no”
$0.45 \leq p \leq 0.54$	“igual de probable que no”
$0.30 \leq p \leq 0.44$	“posible - menos probable que no”
$0.10 \leq p \leq 0.29$	“poco probable”
$0.01 \leq p \leq 0.09$	“muy improbable”
$p < 0.01$	“extremadamente improbable”

Tabla 3.1: Mapeo de probabilidades de acuerdo con la OMM.

3.4.2. Enfoques basados en LM estadísticos

Como se indicó anteriormente, los enfoques estadísticos se basan en las probabilidades extraídas desde un volumen de texto base, ya sea un corpus, anotado o no, texto procedente de la Web, etc. Una de las herramientas primordiales para este tipo de enfoque son los modelos de lenguaje (*LM: Language* Models).

El sistema que propone Vicente et al. (2018) es un ejemplo de *LM* estadístico para la generación de cuentos infantiles en inglés. La tabla 3.2 presenta ejemplos de historias generadas por este método.

Tale (score 3)	**Tale fragment 1 (score 2)**
the two time fly the domestic_fowl	[...]
the domestic_fowl fly the Eden.	the night ride the Moon.
the hare be the companion.	the night state the white hind.
the blind man perform the hare.	the full hour state the pale hyacinth.
the man perform not certain.	the night be the one light.
the man state then dry.	the wing give_birth the peace.
the big discipline snog the hare.	the cold wind give_birth the fire.
Tale (score 3)	**Tale fragment 2 (score 2)**
the sea be the rampart.	[...]
the mighty king look the sea.	the day be the brook.
the ship sail the sea.	the water run the clear.
the sky arrive the wood.	the bright sun travel the water.
the branch look the blue curtain.	the water achieve the bright sunlight.
the bird fly the full thing.	the bright star glitter the water.
the bad idea arrive the expression.	the water induce the thing.
the idea arrive the difficult expression.	the water give_birth the bright thing.
the bird arrive the small son.	the bright thing know the water.
the bad weather give$_b$*irththebird.*	the water state then strange.

Tabla 3.2: Ejemplos de historias de ficción generadas con el enfoque propuesto por Vicente et al. (2018).

3.4.3. Enfoques basados en LM neuronales

Las redes neuronales, también conocidas como redes neuronales artificiales (RNA) o redes neuronales simuladas (SNN), son un subconuunto del aprendizaje automático y están en el centro de los algoritmos de aprendizaje profundo. Su nombre y estructura se inspiran en el cerebro humano, imitando la forma en que las neuronas biológicas se comunican entre si.

Las redes neuronales artificiales (RNA) se componen de capas de nodos, que contienen una capa de entrada, una o más capas ocultas y una capa de salida. Cada nodo, o neurona artificial, se conecta a otro y tiene un peso y un umbral asociados. Si la salida de cualquier nodo individual está por encima del valor umbral especificado, ese nodo se

activa, enviando datos a la siguiente capa de la red. En caso contrario, no se transmite ningún dato a la siguiente capa de la red (Education, 2021).

Redes neuronales recurrentes (RNN)

Una red neuronal recurrente (RNN) es un tipo de red neuronal artificial que utiliza datos secuenciales o de series temporales. Las RNN pasan cada elemento de la secuencia a través de una red de alimentación directa y utilizan la salida del modelo como entrada para el siguiente elemento de la secuencia, lo que permite almacenar la información del paso anterior. En cada iteración, el modelo almacena en su memoria las palabras anteriores encontradas y calcula la probabilidad de la siguiente palabra. Para cada palabra del diccionario, el modelo asigna una probabilidad basada en la palabra anterior, selecciona la palabra con la mayor probabilidad y la almacena en la memoria. La "memoria"de la RNN hace que este modelo sea ideal para la generación de lenguaje, ya que puede recordar el fondo de la conversación en cualquier momento. Sin embargo, a medida que aumenta la longitud de la secuencia, las RNN no pueden almacenar las palabras que se encontraron de forma remota en la frase y hace predicciones basadas sólo en la palabra más reciente. Debido a esta limitación, las RNN son incapaces de producir frases largas coherentes. (Sciforce, 2019).

Long short-term memory (LSTM)

Para abordar el problema de las dependencias de largo alcance, se introdujo una variante de la RNN denominada memoria a corto plazo (LSTM). Aunque son similares a las RNN, los modelos LSTM incluyen una red neuronal de cuatro capas. La LSTM consta de cuatro partes: la unidad, la puerta de entrada, la puerta de salida y la puerta olvidada. Estas permiten a la RNN recordar u olvidar palabras en cualquier intervalo de tiempo ajustando el flujo de información de la unidad. Cuando se encuentra un periodo, la puerta olvidada reconoce que el contexto de la frase puede cambiar y puede ignorar la información del estado actual de la unidad. Esto permite a la red rastrear selectivamente sólo la información relevante, a la vez que minimiza el problema de la desaparición del gradiente, lo que permite al modelo recordar la información durante un periodo más largo.

Aun así, la capacidad de la memoria LSTM está limitada a unos pocos cientos de palabras debido a sus intrínsecamente complejos recorridos secuenciales desde la unidad anterior a la actual. Esta misma complejidad se traduce en unos requisitos computacionales elevados que hacen que los LSTM sean difíciles de entrenar o paralelizar (Sciforce, 2019).

Story Scrambler

Story Scrambler (Pawade and Sakhapara, 2018) es una herramienta de generación automática de texto que produce una nuva historia respecto a algunas historias de entrada previamente proporcionadas. Por ejemplo; el sistema puede ser útil para crear la secuela

de una historia o novela, para replicar el estilo de un poeta o para crear nuevas escenas de una obra de teatro, serie de televisión, etc.

Las diferentes historias que toma la herramienta como entrada pueden tener diferentes líneas argumentales o el mismo argumento. El sistema primero entiende el contexto de cada historia y, después, intenta aprender secuencias de palabras a partir de las múltiples historias de entrada dando lugar a tres módulos: procesamiento de entrada, creación y entrenamiento de la red neuronal y produciendo la nueva historia.

1. Procesamiento de Entrada

   Este primer paso divide los archivos de entrada y almacena palabras únicas en un diccionario.

2. Creacion y Entrenamiento de la Red Neuronal

   Este módulo, crea una red neuronal que nutre dos conjuntos de datos, datosX que contiene palabras, del archivo o archivos de entrada, de una longitud de secuencia particular; datosY que contiene la siguiente palabra correspondiente a la secuencia en datosX. Los datos de datosX se almacenan mapeando las palabras individuales de la secuencia con sus respectivos índices del diccionario generado en el módulo de procesamiento de entrada. A continuación, datosX se remodela y normaliza. El segundo conjunto de datos, se codifica creando una matriz en la que el elemento del índice que corresponda a la palabra respectiva en el diccionario se establece como 1 mientras que los elementos resantes se mantienen como 0. Ahora se define el modelo LSTM. Contará con 256 unidades de memoria (neuronas). Después, se define la densidad.

3. Produciendo la Historia

   Después de entrenar la red, se produce la nueva historia. Se toma como entrada una secuencia aleatoria de semillas considerada como la ventana inicial. La predicción se realiza calculando la probabilidad de las palabas de datosY. Esta probabilidad se calcula utilizando la función de activación *softmax*. A continuación, la ventana se actualiza añadiendo la palabra predicha al final. Una vez asignados los pesos, el programa comienza a generar frases seleccionado palabras, una a una. La primera palabra se selecciona al azar. Las siguientes palabras se deciden an base a la palabra previamente determinada. Al repetir estos pasos varias veces, empiezan a formarse oraciones, párrafos y, en última instancia, una nueva historia.

Las historias formadas evalaudas por humanos obtienen una precisión del 63 %. La precisión del sistema puede mejorarse aún más si se tiene en cuenta el significado de las palabras. También se pueden utilizarse sinónimos para mejorar la precisión.

Input Story 1	"Oh, what a man he is, this Mr. Willy Wonka! cried, Grandpa Joe. 'Did you know, for example, that he has himself invented more than two hundred new kinds of chocolate bars, each with a different center, each far sweeter and creamier and more delicious than anything the other chocolate factories can make! Perfectly true!' cried Grandma Josephine. 'And he sends them to all the four corners of the earth! Isn't that so, Grandpa Joe? It is, my dear, it is. And to all the kings and presidents of the world as well. But it isn't only chocolate bars that he makes. Oh, dear me, no! He has some really fantastic inventions up his sleeve, Mr. Willy Wonka has! Did you know that he has invented a way of making chocolate ice cream so that it stays cold for hours and hours without being in the refrigerator? You can even leave it lying in the sun all morning on a hot day and it won't go runny!"
Input Story 2	"The old man was sitting in the back of the boat and little Charlie Bucket was right beside him. Charlie was holding tightly on to his grandfather's bony old hand. He was in a whirl of excitement. Everything that he had seen so far, the great chocolate river, the waterfall, the huge sucking pipes, the minty sugar meadows, the Oompa-Loompas, the beautiful pink boat, and most of all, Mr Willy Wonka himself had been so astonishing that he began to wonder whether there could possibly be any more astonishments left. Where were they going now? What were they going to see? And what in the world was going to happen in the next room? Isn't it marvelous?' said Grandpa Joe, grinning at Charlie. Charlie nodded and smiled up at the old man. Suddenly, Mr. Wonka, who was sitting on Charlie's other side, reached down into the bottom of the boat, picked up a large mug, dipped it into the river, filled it with chocolate, and handed it to Charlie."
Generated Story	Mr. Willy Wonka was in a whirl of excitement. He had seen so far the four corners of the earth! He has himself invented a way of making chocolate river. He sends them to Oompa-Loompas, the kings and presidents of the world. Isn't it marvelous? cried Grandma Josephine

Figura 3.1: Historia generada a partir de varias entradas de "Charlie y la fábrica de chocolate".

3.5. Transformers

El análisis de la arquitectura Transformer se ha realizado, casi en su totalidas, gracias a el curso presentado por Carrigan et al. (2021).

La arquitectura Transformer se presentó en junio de 2017. La investigación original se centró en las tarea de traducción. A esto le siguió la introducción de varios modelos influyentes, entre ellos:

- Junio de 2018: **GPT**, el primer modelo Transformer preentrenado, utilizado para el ajuste fino en varias tarea de PNL y que obtuvo rersultados de vanguardia.
- Octubre de 2018: **BERT**, es un modelo de preentrenamiento de LC de código abierto desarrollado por investigadores de Google en 20018. Como descendiente directo de GPT, BERT ha superado a varios modelos en LC y ha proporcionado los mejores resultados en pregunta-respuesta (SQuAD v1.1), inferencia de lenguaje natural (MNLI) y otro marcos.
- Febrero de 2019: **GPT-2**, una versión mejorada (y más grande) de GPT que no se hizo pública de inmediato por cuestiones éticas[18].

[18]https://computerhoy.com/noticias/tecnologia/openai-decide-cancelar-generador-texto-ia-porque-demasiado-peligroso-376041

- Octubre de 2019, **DistilBERT**, una versión destilada de BERT que es un 60 % más rápida, un 40 % más ligera en memoria y que sigue conservando el 97 % del rendimiento de BERT.
- Octubre de 2019: **BART** y **T5**, dos grandes modelos que utilizan la misma arquitectura que el modelo original de Transformer.
- Mayo de 2020: **GPT-3**, una versión aún más grande de GPT-2 que es capaz de rendir bien en una variedad de tareas sin necesidad de ajuste fino (llamado aprendizaje de tiro cero o *zero shot* en inglés.

La tabla 3.3 presenta algunos modelos de lenguaje representativos además de los anteriormente mencionados. En esta tabla también se describen otros aspectos: arquitectura, tarea objetivo, corpus de entrenamiento, número de parámetros, etc.).

PTM	Arquitectura	Entrada	Tarea previa al entrenamiento	Corpus	Params	GLUE	FT?
ELMo	LSTM	Texto	BiLM	WikiText-103			No
GPT	Transformer Dec.	Texto	LM	BookCorpus	117M	72.8	Si
GPT-2	Transformer Dec.	Texto	LM	WebText	117M $\sim$ 1542M		No
BERT	Transformer Enc.	Texto	MLM+NSP	WikiEn+BookCorpus	110M $\sim$ 340M	81.9	Si
InfoWord	Transformer Enc.	Texto	DIM+MLM	WikiEn+BookCorpus	$\approx BERT$	81.1	Si
RoBERTa	Transformer Enc.	Texto	MLM	BookCorpus+CC-News +OpenWebText+STORIES	355M	88.5	Si
XLNet	Two-Stream Transformer Enc.	Texto	PLM	WikiEn+BookCorpus+Giga5 +ClueWeb+Common Crawl	$\approx BERT$	90.5	Si
ELECTRA	Transformer Enc.	Texto	RTD+MLM	$\equiv XLNet$	335M	88.6	Si
UniLM	Transformer Enc.	Texto	MLM+NSP	WikiEn+BookCorpus	340M	80.8	Si
MASS	Transformer	Texto	Seq2Seq MLM	Dependiente de la tarea			Si
BART	Transformer	Texto	DAE	$\equiv RoBERTa$	110 % de BERT	88.4	Si
T5	Transformer	Texto	Seq2Sq MLM	Colossal Clean Crawled Corpus (C4)	220M $\sim$ 11B	89.7	Si
ENRIE(THU)	Transformer Enc.	Texto+Entidades	MLM+NSP+dEA	WikiEn+WikiData	114M	79.6	Si
KnowBERT	Transformer Enc.	Texto	MLM+NSP+EL	WikiEn+WordNet/Wiki	253M $\sim$ 523M		Si
K-BERT	Transformer Enc.	Texto+Triples	MLM+NSP	WikIZh+WebtextZh+CN-DBpedia +HowNet+MedicalKG	$\approx BERT$		Si
KEPLER	Transformer Enc.	Texto	MLM+KE	WikiEn+Wikidata/WordNet			Si
WKLM	Transformer Enc.	Texto	MLM+ERD	WikiEn+Wikidata	$\approx BERT$		Si

Tabla 3.3: Modelos Pre-entrenados (PTM) más representativos (Qiu et al., 2020).

Todos los modelos Transformer mencionados anteriormente (GPT, BERT, BART, T5, etc.) han sido entrenados como modelos lingüísticos. Esto significa que se han entrenado con grandes cantidades de texto en bruto de forma autosupervisada. El aprendizaje autosupervisado es un tipo de entrenamiento en el que el objetivo se calcula automáticamente a partir de las entradas del modelo. Esto significa que no se necesitan humanos para etiquetar los datos.

Este tipo de modelo desarrolla una comprensión estadística del lenguaje sobre el que se ha entrenado, pero no es muy útil para tareas prácticas específicas. Por eso, el modelo general preentrenado pasa por un proceso llamado aprendizaje de transferencia. Durante este proceso, el modelo se perfecciona de forma supervisada — es decir, utilizando etiquetas anotadas por humanos — en una tarea determinada.

Un ejemplo de tarea es predecir la siguiente palabra de una frase tras haber leído las n palabras anteriores. Esto se llama modelización causal del lenguaje porque el resultado depende de las entradas pasadas y presentes, pero no de las futuras.

Aparte de algunos casos atípicos (como el de DistilBERT), la estrategia general para lograr un mejor rendimiento consiste en aumentar el tamaño de los modelos, así como

la cantidad de datos con los que están preentrenados.

Desgraciadamente, el entrenamiento de un modelo, sobre todo si es grande, requiere una gran cantidad de datos.

Por eso es fundamental compartir los modelos lingüísticos: compartir los pesos entrenados y construir sobre los pesos ya entrenados reduce el coste global de computación y la huella de carbono de la comunidad.

3.5.1. Aprendizaje por transferencia

El preentrenamiento es el acto de entrenar un modelo desde cero: los pesos se inicializan aleatoriamente y el entrenamiento comienza sin ningún conocimiento previo.

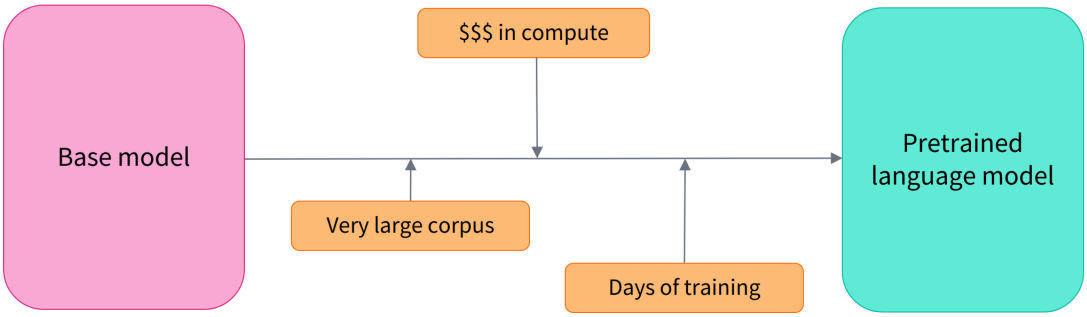

Figura 3.2: *Training from scratch* (entrenamiento desde cero)

Este preentrenamiento suele realizarse sobre cantidades muy grandes de datos. Por lo tanto, requiere un corpus de datos muy grande, y el entrenamiento puede durar hasta varias semanas.

El ajuste fino, en cambio, es el entrenamiento que se realiza después de haber preentrenado un modelo. Para realizar el ajuste fino, primero se adquiere un modelo lingüístico preentrenado y luego se realiza un entrenamiento adicional con un conjunto de datos específico para la tarea. ¿Por qué no se entrena directamente para la tarea final? Hay un par de razones:

- El modelo preentrenado ya fue entrenado en un conjunto de datos que tiene algunas similitudes con el conjunto de datos de ajuste. De este modo, el proceso de ajuste puede aprovechar los conocimientos adquiridos por el modelo inicial durante el preentrenamiento (por ejemplo, en el caso de los problemas de PNL, el modelo preentrenado tendrá algún tipo de conocimiento estadístico del lenguaje que se utiliza para la tarea).
- Dado que el modelo preentrenado ya fue entrenado con muchos datos, el ajuste fino requiere muchos menos datos para obtener resultados decentes.
- Por la misma razón, la cantidad de tiempo y recursos necesarios para obtener buenos resultados es mucho menor.

Por ejemplo, se puede aprovechar un modelo preentrenado en el idioma inglés y luego afinarlo en un corpus arXiv, lo que daría como resultado un modelo basado en la ciencia/investigación. El ajuste sólo requerirá una cantidad limitada de datos: el conocimiento que el modelo preentrenado ha adquirido se "transfiere", de ahí el término aprendizaje por transferencia.

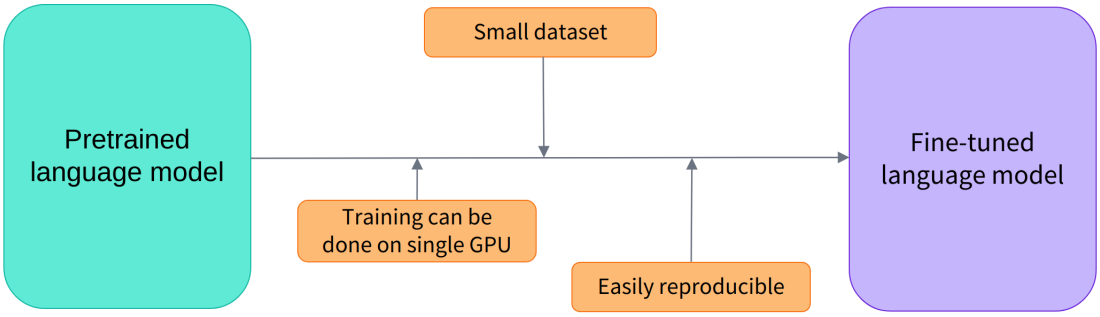

Figura 3.3: *Fine tunning* (ajuste fino)

El ajuste fino de un modelo tiene, por tanto, menores costes de tiempo, datos, financieros y ambientales. También es más rápido y fácil iterar sobre diferentes esquemas de ajuste fino, ya que el entrenamiento es menos restrictivo que un preentrenamiento completo.

Este proceso también permite obtener mejores resultados que el entrenamiento desde cero (a no ser que se disponga de muchos datos), por lo que siempre hay que intentar aprovechar un modelo preentrenado -uno lo más parecido posible a la tarea que se tiene entre manos- y afinarlo.

3.5.2. Arquitectura general

En esta sección se presenta la arquitectura general de la arquitectura Transformer. El modelo se compone principalmente de dos bloques:

- **Encoder:** El codificador recibe una entrada y construye una representación de la misma (sus características). Esto significa que el modelo está optimizado para adquirir la comprensión de la entrada.
- **Decoder:** El decodificador utiliza la representación del codificador (características) junto con otras entradas para generar una secuencia objetivo. Esto significa que el modelo está optimizado para generar salidas.

Cada una de estas partes puede utilizarse de forma independiente, en función de la tarea:

- **Modelos sólo de codificador:** Buenos para tareas que requieren la comprensión de la entrada, como la clasificación de frases y el reconocimiento de entidades

con nombre. Los representantes de esta familia de modelos son: ALBERT, BERT, DistilBERT, ELECTRA, RoBERTa.

- **Modelos sólo de decodificación:** Buenos para tareas generativas como la generación de textos. Los representantes de esta familia de modelos son: CTRL, GPT, GPT-2, Transformer XL.

- **Modelos de codificador-decodificador o de secuencia a secuencia:** Buenos para tareas generativas que requieren una entrada, como la traducción o el resumen Los representantes de esta familia de modelos son: BART, mBART, Marian, T5.

3.6. Herramientas de GLN

En este apartado se especifican algunas herramientas similares al sistema propuesto para la generación de lenguaje natural (GLN).

Existen apicaciones como Shelley[19]. Esta herramienta es capaz de generar relatos de horror colaborando con los usuarios a través de Twitter. Scherezade-IF[20] y AI-Dungeon[21], son herramientas colaborativas al igual que Shelley. Estas interactúan con los usuarios por medio de una interfaz para generar historias de ficción. Magic AI[22], al contrario que las herramientas anteriores, es la que más se asemeja a el sistema propuesto. Esta herramienta es capaz de generar descripciones de cartas Magic y utiliza el modelo GPT-2.

3.6.1. Shelley

Shelley es una inteligencia artificial capaz de escribir relatos de terror en colaboración con humanos. Se trata de una tecnología basada en el aprendizaje profundo (deep learning).

La red está entrenada con una colección 150.000 relatos, encontrados en Nosleep del agregador de noticas de Reddit, y clásicos de autores como Edgar Alan Poe y Sthepen King. Esta aplicación, crea sus propios relatos detectando patrones narrativos en las historias, con una extensión limitada de cuatro o cinco párrafos.

En colaboración con los usuarios de Twitter, la aplicación es capaz de aprender y mejorar su inteligencia. Con la frecuencia de aproximadamente una hora lanza un nuevo hilo con el que inicia una historia lista para ser continuada por cualquier persona (figura 3.4) . Solo hay que contestar aquel tuit que termine con la etiqueta #youturn y seguir unas sencillas normas:

- Responder con un máximo de tres tuits (con el formato adecuado según Twitter).

[19] https://retina.elpais.com/retina/2017/10/30/tendencias/1509381898_152636.html
[20] https://www.xataka.com/robotica-e-ia/este-algoritmo-genera-historias-del-estilo-elige-tu-propia-aventura-de-duracion-ilimitada
[21] https://www.neoteo.com/ai-dungeon-aventuras-de-texto-infinitas-con-inteligencia-artificial/
[22] https://minimaxir.com/apps/gpt2-mtg/

- Finalizar el tuit o hilo de respuesta con la etiqueta #youturn -si lo que quieres es que continúe la historia- o con la etiqueta #theend para terminarla.
- Contestar al tuit de cada hilo si se desea continuar dicho hilo.

El sistema no responde siempre. Importa el contenido del tuit y si ha otras personas les ha gustado (mediante me gusta y retuits). Además, esta diseñado para evitar por ejemplo que responda a mensajes incoherentes, racistas, pornográficos, etc.

Por el momento, la herramienta no es capaz de profundizar en sus textos. Según afirma Cebrián, uno de los responsables de su creación: "Puede describir muy bien un monstruo pero no se le da muy bien crear suspense, seguir una trama y proponer una revelación". La interacción con los usuarios de Twitter se cree que podría mejorar la calidad de los textos.

Figura 3.4: Relato escrito por Shelley en colaboración con un usuario de Twitter.

3.6.2. Scheherazade-IF

Scheherazade-IF (Guzdial et al., 2015) es un sistema narrativo que genera historias interactivas de ficción de situaciones comunes. Esto es gracias a la colaboración colectiva o Crowdsourcing, de manera que son los propios colaboradores humanos los que la nutren de historias y situaciones.

Los colaboradores humanos interactúan con el sistema de tres maneras: como jugadores, miembros de la multitud o "autores" . Los "autores" piden a Scheherezade-IF que construya una narrativa ficticia sobre un tema o situación específica. Si el sistema no tiene un modelo de la situación en su memoria, busca aprender de la multitud. Esto produce un corpus altamente especializado de narrativas de ejemplo a partir de las cuales se obtiene un modelo general del tema, conocido como *ploth graph* o gráfico de trama. La compilación de ejemplos es un paso clave que permite la ejecución interactiva; las historias se entretejen de manera que en cualquier momento, el jugador que tenga alternativas para elegir y consecuencias razonables.

Una vez que se recopilan las historias de ejemplo, el sistema debe hacer dos cosas para compilar un gráfico de trama: (1) debe determinar cuáles son los eventos primitivos para cada situación dada, y (2) debe determinar la estructura del gráfico identificando el orden de los eventos y las exclusiones mutuas. Los eventos son grupos de frases de diferentes ejemplos que se refieren semánticamente a la misma actividad.

A partir de los eventos, Scheherezade-IF genera la estructura de un gráfico de trama identificando las relaciones de precedencia y las exclusiones mutuas.

Para presentar opciones al jugador, el sistema determina primero qué eventos de la trama son ejecutables. Un evento es ejecutable cuando todos sus predecesores directos, no opcionales, se han ejecutado, excepto aquellos padres excluidos por relaciones de exclusión mutua. En particular, al principio del gráfico de la trama "atraco al banco" en la figura 4, si el jugador estuviera jugando como Juan, los eventos ejecutables serían 1, 2 y 3, ya que son los únicos eventos sin predecesores y se presentarían al jugador. El jugador podría entonces elegir el evento de la trama 3 "Juan cubre la cara". Al elegir esta opción se envía un mensaje al sistema para que devuelve una descripción de texto del evento, que luego se muestra al jugador.

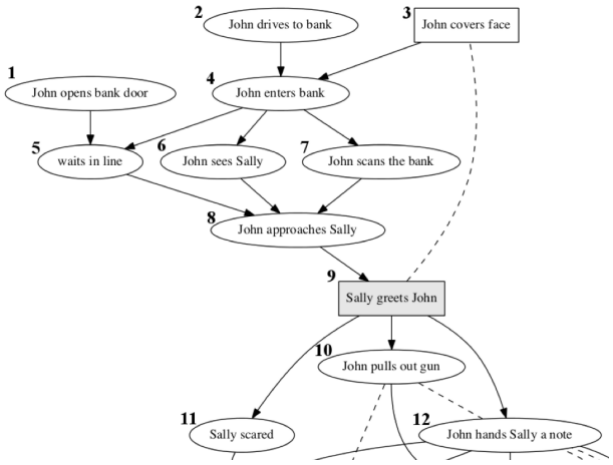

Figura 3.5: Una subsección del gráfico de trama de atraco al banco.

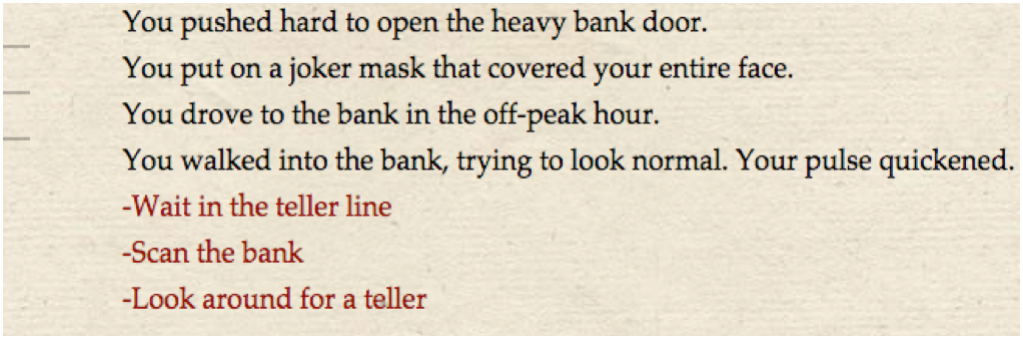

Figura 3.6: Captura de pantalla de la reproducción de Scheherazade-IF.

3.6.3. AIDungeon

AIDungeon es un juego de aventura de texto en el cual los jugadores pueden elegir un género para su historia (por ejemplo, fantasía, misterio, apocalíptico o zombies); sin embargo. a diferencia de las aventuras tradicionales que usan contenido predeterminado, el sistema usa su inteligencia artificial para generar historias ilimitadas y personalizadas.

Hay tres métodos de interacción principales que pueden ser usados para le entrada del texto del usuario:

- Hacer: Tiene que estar seguido por un verbo, permitiendo al jugador realizar una acción.
- Decir: Tiene que estar seguido por frases de diálogo, permitiendo a los jugadores comunicarse con otros personajes.
- Historia: Puede estar seguido por frases describiendo algo que pasa para progresar la historia, o que los jugadores quieren que la IA sepa para futuras acciones.

También se puede proporcionar una entrada de texto vacia para que la IA genere más contenido.

La IA del juego fue entrenada aproximadamente con 30 megabytes de contenido de chooseyourstory.com (un sitio web comunitario de contenido inspirado en librojuegos, escrito pro colaboradores de distintos niveles), permitiéndole generar sus propias aventuras originales y darle respuestas complejas a la entrada del usuario.

Sin embargo, en noviembre de 2019, OpenAI liberó el modelo GPT-2 de 1.5 mil millones de parámetros. El juego fue actualizado para funcionar con esta nueva versión del modelo.

En julio de 2020, los desarrolladores mejoraron el modelo para utilizar GPT-3 con el API de OpenAI.

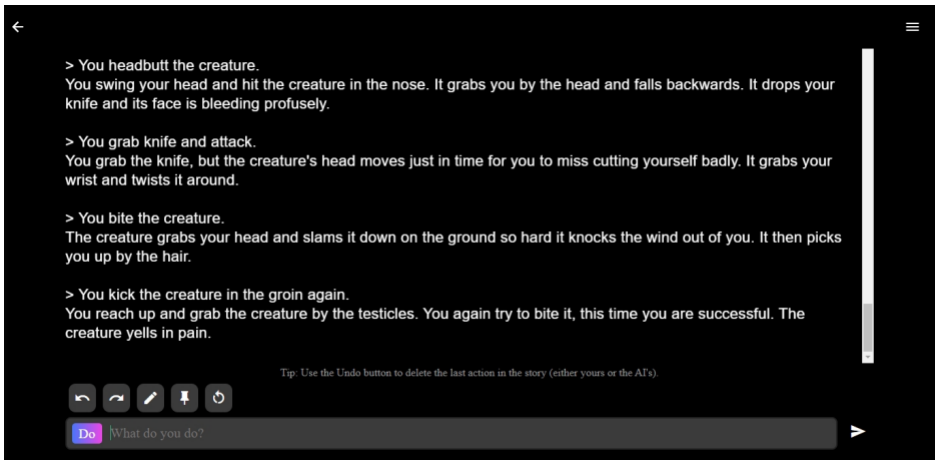

Figura 3.7: Historia interactiva generada por el juego AIDungeon.

4 Metodología

En este capítulo se especifica como se ha llevado a cabo el estudio de la motivación tras este trabajo, la generación de lenguaje natural (GLN). Además, se exponen las tecnologías y herramientas necesarias para desempeñar esta labor.

Vicente et al. (2015) presenta un análisis del estado actual de la GLN indispensable para entender los fundamentos tras el area de la GLN. El análisis realizado por Vicente et al. (2015), junto a otras fuentes de información (Vaca, 2021) (Carrigan et al., 2021), son fundamentales para llevar a cabo el estudio de la evolución del GLN natural que se presenta en la sección 3.3. Esta sección, la cual hace un recorrido de los distintos enfoques para abordar la tarea de la GLN, desde los más tradicionales, hasta el estado actual del arte, los Transformer, ha permitido concluir las tecnologías idoneas para abordar la tarea de la GLN. Estas tencologías son la arquitectura Transformer y los modelos de lenguaje. El modelo de lenguaje que utiliza el sistema propuesto en este trabajo es GPT-2.

4.1. Tecnologías

- **Python[23]:** Python es un lenguaje de programación interpretado cuya filosofía hace hincapié en la legibilidad de su código. Se trata de un lenguaje de programación multiparadigma, ya que soporta parcialmente la orientación a objetos, programación imperativa y, en menor medida, programación funcional. Es un lenguaje interpretado, dinámico y multiplataforma. La versión utilizada es la 3.7.8.

- **TensorFlow[24]:** TensorFlow es una biblioteca de código abierto para aprendizaje automático a través de un rango de tareas, y desarrollado por Google para satisfacer sus necesidades de sistemas capaces de construir y entrenar redes neuronales para detectar y descifrar patrones y correlaciones, análogos al aprendizaje y razonamiento usados por los humanos.

- **GPT-2[25]:** Generative Pre-trained Transformer 2 (GPT-2) es una inteligencia artificial de código abierto creada por OpenAI en febrero de 2019. GPT-2 traduce texto, responde a preguntas, resume pasajes, y genera salidas de texto a un nivel que, si bien a veces es indistinguible del de los humanos, puede volverse repetitivo o sin sentido cuando genera pasajes largos.

[23] https://www.python.org/
[24] https://www.tensorflow.org/
[25] https://openai.com/blog/better-language-models/

- **Transformer[26]:** Un Transformer es un modelo de aprendizaje profundo que adopta el mecanismo de la atención, sopesando diferencialmente la importancia de cada parte de los datos de entrada. Se utiliza principalmente en el campo del procesamiento del lenguaje natural (PLN) y en la visión por ordenador.

4.2. Herramientas

- **Overleaf[27]:** Editor online para elaborar la documentación del TFG.
- **DeepL Traductor[28]:** Traductor online para traducir los textos del corpus recolectado.

[26]https://arxiv.org/pdf/1706.03762.pdf
[27]https://es.overleaf.com/project
[28]https://www.deepl.com/es/translator

5 Hipótesis y enfoque propuesto

A lo largo de los capítulos anteriores se ha dejado entrever el propósito del sistema que se plantea en este trabajo. Como bien sabemos, el sistema pretende generar descripciones de forma automática asociadas a juegos de rol y en español. Por lo cual, para llevar a cabo tal cometido el enfoque propuesto se basa en la arquitectura Transformer, concretamente en el *LM* GPT-2. Este escenario suscita varios interrogantes: ¿Por qué se ha elegido este propósito?, ¿Por que se ha escogido la arquitectura Transformer y el modelo GPT-2?, y por último, ¿Por qué el idioma escogido es el español?. Este capítulo pretende resolver ambos interrogantes y proponer una arquitectura para resolver el enfoque propuesto.

5.1. Justificación

¿Por qué el propósito del sistema es generar textos asociados a juegos de rol?

Un juego de rol (*RPG: role-playing game*) es un juego en el que uno o más jugadores representan un determinado rol, papel o personalidad. Estos juegos están caracterizados por la minuciosidad de los detalles del mundo que los rodea. Es así, a causa de la necesidad de hacer sentir al jugador de que el mundo que lo rodea es real y esta viviendo una aventura. Los juegos más relevantes de esta temática son: World of Warcraft[29], The Witcher[30], Skyrim[31], etc. Todos estos juegos tienen en común que el mundo que albergan tiene una historia detrás. Es decir, la geografía, la fauna y la flora, las ciudades, etc. albergan el significado de su existencia. Cada pequeño detalle tiene su historia.

Evidentemente estas historias no se generan solas. Son el trabajo de un equipo de profesionales. Ahí es donde entra el sistema propuesto. Este sistema aspira a ser una herramienta que facilite la labor de esos expertos generando textos que doten de significado a diferentes elementos de ese tipo de juegos como: estructuras, flora, fauna, comida, bebida (como en este caso), etc. Naturalmente, el objetivo del sistema no es generar la historia tras el universo del juego, lo cual es una tarea sumamente compleja, pero si de pequeños elementos de ese mundo, universo o de lo que se trate.

¿Por qué se ha escogido la arquitectura Transformer y el modelo GPT-2?

Básicamente, la elección de estas dos tecnologías son fruto de la investigación realizada en el capítulo 3. Por un lado, el enfoque basados en modelos de lenguaje neuronales y la arquitectura Transformer son las tecnologías de vanguardia que mejores resultados están

[29] https://worldofwarcraft.com/es-es/
[30] https://es.wikipedia.org/wiki/The_Witcher_$serie_de_videojuegos$
[31] https://elderscrolls.bethesda.net/es/skyrim

ofreciendo para diversas tareas de la LC, las cuales comprenden la GLN. En relación con GPT-2, tiene una arquitectura Transformer Decoder. Esta arquitectura es la que mejores resultados ofrece para la tarea de generación de lenguaje natural. Las arquitecturas Transformer Encoder y Transformer Encoder-Decoder están destinadas a otro tipo de tareas de la LC (3.5).

¿Por qué el idioma escogido es el español?

El motivo principal tras esta elección es la escasez de recursos en otros idiomas diferentes del inglés para abordar la tarea de la GLN. Por ejemplo, al realizar una búsqueda de modelos GPT-2 en español en Hugginface[32] (comunidad que alberga miles de recursos para la tarea de la GLN) aparecen únicamente 3 resultados de los 377 modelos.

Este escenario no ocurre únicamente con modelos de lenguaje preentrenados, sino también con otras herramientas como: corpus, artículos, APIs, etc.

Por eso, este trabajo permite aportar su granito de arena aportando un modelo de lenguaje diferente del inglés.

5.2. Arquitectura

En esta sección se presenta la arquitectura planteada para abordar el sistema. Únicamente se plantea el funcionamiento de cada sección, la impementación se ejecutará en el capítulo siguiente.

La arquitectura que se propone consta de tres fases:

1. Preparación de los datos
2. Entrenamiento del modelo de lenguaje neuronal
3. Generación de texto

La figura 5.1 ilustra un diagrama de flujo de las fases del sistema.

[32] https://huggingface.co/

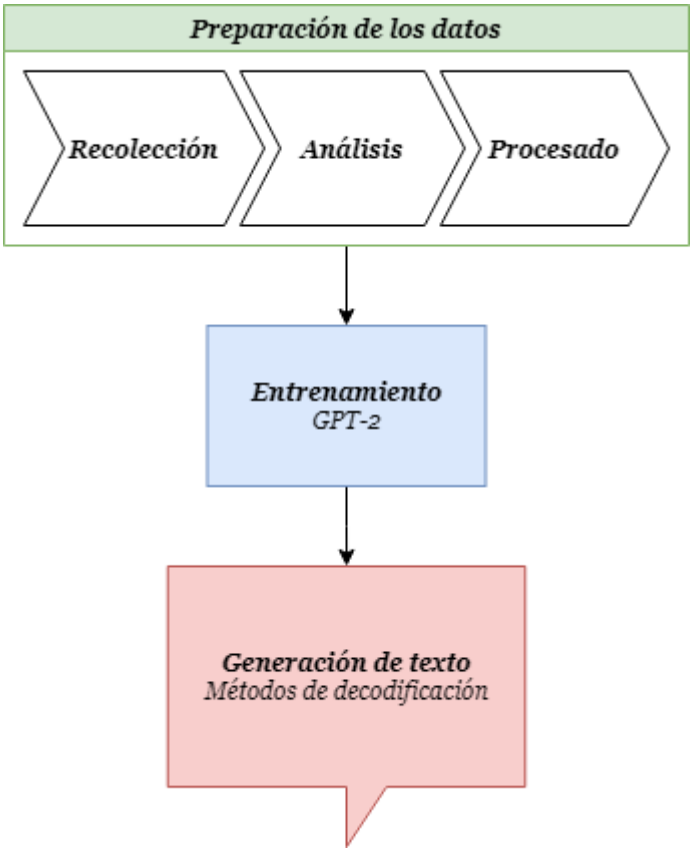

Figura 5.1: Diagrama de flujo de las fases del sistema.

Preparación de los datos

Esta fase consta de tres etapas:

1. **Recolección:** En esta etapa se recolectan los datos en bruto. Los datos recolectados en este periodo no son los datos de entrenamiento. Estos datos tienen que transitar otras dos etapas para considerarse datos de entrenamiento.

2. **Análisis:** En esta segunda etapa se analiza el contenido de los datos recolectados. El contenido de los datos no es válido en su totalidad. El objetivo de esta fase es depurar esos datos de forma que alberguen únicamente la información necesaria.

3. **Procesado:** Esta fase es la que requiere mayor inversión de tiempo. En esta fase se traduccen los datos recolectados del inglés a español y se corrigen los posibles fallos de traducción.

Entrenamiento

En esta sección se comenta como se pretende llevar a cabo el entrenamiento. En el capítulo 3.5 se señala que las técnicas más relevantes para entrenar los modelos de lenguaje son dos:

- *Training from scratch* o entrenamiento desde cero, y
- *Fine tunning* o ajuste fino

Según la investigación realizada, y los artículos consultados, la técnica que mejores resultados ofrece es el ajuste fino. Aún así, esta técnica presenta dos invonvenientes para el sistema planteado: el dominio del sistema es demasiados específico (descripciones de bebidas asociadas a juegos de rol) y los recuros en español escasean.

Ciertamente, es posbile aplicar el ajuste fino en un modelo GPT-2 en inglés para otro idioma, como el portugués[33], o el español[34]. La diferencia con el ajuste fino habitual, para modelos con el mismo idioma, es que hay que reentrenar el vocabulario del modelo en el idioma interesado.

En tal caso, haciendo alusión a la limitación de recursos en español y especialización del dominio, se pretende entrenar el modelo de lenguaje desde cero. Cabe resaltar, que esta decisión también busca analizar la calidad del texto generado con este tipo de entrenamiento.

Generación de texto

Para generar la salida del texto se utilizan distitns métodos de decodificación que se verán en el capítulo siguiente.

[33] https://medium.com/pierre_guillou/faster-than-training-from-scratch-fine-tuning-the-english-gpt-2-in-any-language-with-hugging-f2ec05c98787

[34] https://huggingface.co/datificate/gpt2-small-spanish

6 Desarrollo

En este capítulo se implementa la arquitectura planteada en la sección 5.2 para abordar el sistema.

6.1. Preparación de los datos

En esta sección se describe como se ha llevado a cabo cada una de las etapas de la fase de preparación de los datos: recolección, análisis y procesado.

Las tareas de recolectar un corpus, analizarlo y procesarlo puden parecer triviales, sin embargo, estas tareas, junto a los factores del idioma (español) y a la restricción del dominio (generar textos de bebidas asociadas a juegos de rol), han resultado en una fase laboriosa y con una inversión de tiempo sumamente superior a la de las fases posteriores.

6.1.1. Recolección

Esta primera etapa, como se menciona en la especificación de la arquitectura (sección 5.2, recolecta los datos en bruto. La tabla 6.1 presenta las webs en las que se han recolectado los datos para formar el corpus. En esa misma tabla también aparece el número de descripciones (de 20 a 50 palabras aprox.) de bebidas recogidas de cada web. Estas webs ofrecen recursos a la comunidad de jugadores del juego de mesa D&D[35] para elaborar sus partidas de rol.

Corpus	Descripciones
THE SWORD & TORCH IN[36]	100
DNDSPEAK[37]	100
AURICAN'S LAIR[38]	100

Tabla 6.1: Corpus de descripciones de bebidas asociadas a juegos de rol.

6.1.2. Análisis

En esta etapa se depura el corpus de modo que albergue únicamente la información necesaria. En este caso, algunas descripciones incluían un fragmento que describía los efectos producidos en el jugador que consumiese la bebida descrita. Por ejemplo:

> "If a drinker fails their saving throw, they will be paralyzed from the neck down for 1D20 minutes."

[35] https://dnd.wizards.com/

Esta descripción hace alusión al tradicional sistema de dados del cual hace uso este tipo de juegos. En función del resultado en una tirada de n dados, el efecto de parálisis perdura más o menos tiempo. Los fragmentos que albergaban este tipo de alusiones a la mecánica de dados se han alterado o eliminado para que el sistema no haga referencia a este tipo de mecánica. Esta mecánica es frecuente en los juegos de rol de mesa, pero no en los juegos de rol de ordenador, en los que se centra este sistema.

6.1.3. Procesado

En esta última etapa se traducen los datos para elaborar el corpus final. Para esta tarea se ha utilizado la herramienta DeepL Translate[39]. En algunos casos las traducciones generadas presentaban errores gramaticales que requerían correción. Además, se añaden al principio y final de cada descripción las etiquetas especiales < |startoftext| > y < |endoftext| >.

6.2. Entrenamiento

Esta fase se ha llevado a cabo siguiendo las directrices de un artículo de la revista digital científica *towards data sciencie*[40] que entrena un modelo GPT-2 desde cero para el Bengali[41].

6.2.1. Recolección de lo datos

Esta fase se ha llevado a cabo en la sección 6.1.1

6.2.2. Tokenización

En esta fase se ha empleado un tokenizador[42] de codificación de pares de bytes (BPE). Este tokenizador separa el texto en subpalabras con fin de no tratar diferentes forma de la palabra como diferentes (por ejemplo, *greatest* se tratará como dos tokens: *great* y *est*, lo que es ventajoso ya que conserva la similitud entre *great* y *greatest*, mientras que *greatest* tiene otro token *est* añadido que lo hace diferente. Además, no es un nivel tan bajo como la codificación a nivel de caracteres, que no conserva ningún valor de una palabra concreta.

[39] https://www.deepl.com/es/translator
[40] https://towardsdatascience.com/
[41] https://towardsdatascience.com/train-gpt-2-in-your-own-language-fc6ad4d60171
[42] La tokenización es una forma de separar un texto en unidades más pequeñas llamadas tokens. Los tokens pueden ser palabras, caracteres o subpalabras. Por lo tanto, la tokenización puede clasificarse en tres tipos: tokenización de palabras, de caracteres y de subpalabras (de n-gramas).

```python
class BPE_token(object):
    def __init__(self):
        self.tokenizer = Tokenizer(BPE())
        self.tokenizer.normalizer = Sequence([
            NFKC()
        ])
        self.tokenizer.pre_tokenizer = ByteLevel()
        self.tokenizer.decoder = ByteLevelDecoder()

    def bpe_train(self, paths):
        trainer = BpeTrainer(vocab_size=50257, show_progress=True, inital_alphabet=ByteLevel.alphabet(), special_tokens=[
            "<|startoftext|>",
            "<pad>",
            "<|endoftext|>",
            "<unk>",
            "<mask>"
        ])
        self.tokenizer.train(paths,trainer)

    def save_tokenizer(self, location, prefix=None):
        if not os.path.exists(location):
            os.makedirs(location)
        self.tokenizer.model.save(location, prefix)
```

Figura 6.1: Clase BPE.

La figura 6.2 presenta los resultados de la tokenización. El total de palabras tokenizadas es de 3625.

Figura 6.2: Palabras tokenizadas.

6.2.3. Inicialización del modelo

En esta etapa se inicializa el modelo. En primer lugar es necesario cargar el vocabulario generado del proceso de tokenización en el tokenizador para el modelo GPT-2. La figura 6.3 presenta como se carga el vocabulario tras el proceso de tokenización. En esta figura también podemos ver que a los tokens *eos_token* y *bos_token* se les asignan dos etiquetas especiales, $<$ |startoftext| $>$ y $<$ |endoftext| $>$. Estas etiquetas permiten al modelo aprender la estructura del texto que se pretende generar (Dittmar, 2021). Cabe señalar, que estas etiquetas se añaden al corpus en la sección 6.1.3.

```
save_path = './Desktop/TFG_Test/token_posada2'
paths = [str(x) for x in Path("./Desktop/TFG_Test/corpus_drinks/").glob("**/*.txt")]

# Loading tokenizer from the saved model path
tokenizer = GPT2Tokenizer.from_pretrained(save_path)

tokenizer.add_special_tokens({
  "eos_token": "<|startoftext|>",
  "bos_token": "<|endoftext|>",
  "unk_token": "<unk>",
  "pad_token": "<pad>",
  "mask_token": "<mask>"
})

# Creating the configurations from which the model can be made
config = GPT2Config(
  vocab_size=tokenizer.vocab_size,
  bos_token_id=tokenizer.bos_token_id,
  eos_token_id=tokenizer.eos_token_id
)
# Creating the model
model = TFGPT2LMHeadModel(config)
```

Figura 6.3: Cargar vocabulario tokenizado.

El siguiente paso es codificar toda la cadena en un solo string.

```
single_string = ''
for filename in paths:
  with open(filename, "r", encoding='utf-8') as f:
    x = f.read()
    single_string += x
string_tokenized = tokenizer.encode(single_string)
```

Figura 6.4: Codificar cadena.

Después de haber codificado toda la cadena, se hace un conjunto de datos de TensorFlow, cortando los datos en intervalos iguales para que el modelo pueda aprender. La figura 6.5 presenta el código que ejecuta esta acción.

```
examples = []
block_size = 50
BATCH_SIZE = 12
BUFFER_SIZE = 1000
for i in range(0, len(string_tokenized) - block_size + 1, block_size):
  examples.append(string_tokenized[i:i + block_size])
inputs, labels = [], []

for ex in examples:
  inputs.append(ex[:-1])
  labels.append(ex[1:])
dataset = tf.data.Dataset.from_tensor_slices((inputs, labels))
dataset = dataset.shuffle(BUFFER_SIZE).batch(BATCH_SIZE, drop_remainder=True)
```

Figura 6.5: Hacer un conjunto de datos TensorFlow.

6.2.4. Entrenamiento del modelo

Ahora se define el optimizador, las funciones de pérdida y las métricas, y empieza el entrenamiento.

```
# defining our optimizer
optimizer = tf.keras.optimizers.Adam(learning_rate=3e-5, epsilon=1e-08, clipnorm=1.0)
# defininig our loss function
loss = tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True)
# defining our metric which we want to observe
metric = tf.keras.metrics.SparseCategoricalAccuracy('accuracy')
# compiling the model
model.compile(optimizer=optimizer, loss=[loss, *[None] * model.config.n_layer], metrics=[metric])

num_epoch = 100

checkpoint_path = "./Desktop/TFG_Test/cp_posada/cp_cpkt"

model_checkpoint_callback = tf.keras.callbacks.ModelCheckpoint(
    filepath=checkpoint_path,
    save_weights_only=True,
    monitor='logits_accuracy',
    mode='max',
    save_best_only=True,
    verbose=1)

model.fit(dataset,  epochs=num_epoch, callbacks=[model_checkpoint_callback])
```

Figura 6.6: Definición de funciones de pérdida, métricas y entrenamiento del modelo.

6.3. Generación de texto

Los resultados del modelo se presenta y evaluan en el capítulo 7.

7 Evaluación

En este capítulo se evalua la calidad del texto generado por el modelo entrenado. Esta labor se lleva a cabo testeando el texto generado con dos técnicas de decodificación distintas *Beam Search (b=16)* y *Nucleus (p=0.95)*.

Estas técnicas y valores se han elegido haciendo caso al valor de perplejidad que presentan. Por un lado *Beam Search (b=16)* exhibe una perplejidad de 1.48. Por el contrario, *Nucleus (p=0.95)* presenta una perplejidad de 13.13, semejante a la perplejidad humana (Holtzman et al., 2020).

Según Wikipedia (2021):

> "En la teoría de la información, la perplejidad es una medida de lo bien que una distribución de probabilidad o un modelo de probabilidad predice una muestra. Puede utilizarse para comparar modelos de probabilidad. Una perplejidad baja indica que la distribución de probabilidad es buena para predecir la muestra."

Haciendo caso a la definición de Wikipedia (2021) de la perplejidad se deduce que la técnica de decodificación *Nucleus (p=0.95)* al tener un valor de perplejidad semejante al del ser humano, tiene una mayor capacidad de creatividad que el método de decodificación *Beam Search (b=16)*. El lenguaje natural no maximiza la probabilidad (Holtzman et al., 2020).

La tabla 7.1 presenta ejemplos de la salida generados a partir de un texto de entrada con la técnica de decodificación *Beam Search (b=16)*.

Entrada	Salida
"Una bebida cara"	"Una bebida cara, ya que se dice que beber una pinta de ésta es tan bueno como comer una pequeña comida y es bastante"
"Esta sidra"	"Esta sidra se convierte en una bebida negra y crujiente con claras notas de arándanos y un aroma terroso."
"Un whisky sabroso"	"Un whisky sabroso, pero con un sabor agrio."

Tabla 7.1: Ejemplos de textos generados con la técnica de decodificación *Beam Search (b=16)*.

Al contrario, la tabla 7.2 expone los textos generados con la técnica de decodificación *Nucleus (p=0.95)* a patir de la entrada de la primera columna.

Entrada	Salida
"Una bebida cara"	"Una bebida cara, sino que te deja si se bebe a los nigromantes."
"Esta sidra"	"Esta sidra se elabora en verano con leche de urogallo mezclada de invierno. El sabor de miel, tiene un sabor fuerte y se"
"Un whisky sabroso"	"Un whisky sabroso, pero no demasiado por la gente que tiene un regusto."

Tabla 7.2: Ejemplos de textos generados con la técnica de decodificación *Nucleus (p=0.95)*.

Las salidas que presentan una mejor calidad lingüística son los textos generados con la técnica de decodificación *Beam Search (b=16)*. A pesar de no conseguir malos resultados, los textos generados por la técnica *Nucleus (p=0.95)* manifiestan cierta incoherencia.

Este escenario es evidente debido a la escasez de datos de entrenamiento del modelo. Las técnica con reducida perplejidad (mejor distribución de probabilidad para predecir la muestra) se ajustan a los datos de entrenamiento produciendo mejores resultados en un vocabulario reducido como es el caso de este sistema. Pero, como indica Holtzman et al. (2020), el lenguaje natural no maximiza la probabilidad. Así, a pesar de los resultados producidos por la técnica *Beam Search (b=16)*, podría considerarse que el sistema no genera verdadero LN, es decir, que no es creativo.

Por otro lado, los textos generados por la técnica *Nucleus (p=0.95)*, pretenden ser creativos dada su alta perplejidad, y a pesar de conseguirlo (la salida no se asemeja a los datos de entrenamiento, como anteriormente), presentan cierta incoherencia.

8 Conclusiones

En este trabajo final de grado (TFG) se ha propuesto y desarrollado un sistema de GLN para generar descripciones asociadas a juegos de rol en español. El enfoque tras este sistema esta fundamentado en la arquitectura Transfomer y el modelo GPT-2.

Las decisiones del propósito y enfoque del sistema se han tomado tras una minuciosa investigación del campo de la GLN. Por un lado, la arquitectura Transformer es la tecnología de vanguardia para abordar la tarea de la GLN, así como otros campos de la LC. Además, GPT-2 es uno de los modelos de lenguaje que mejores resultados ofrece para la GLN. Respecto al propósito del sistema, los juegos de rol requieren una descripción minuciosa del entorno que los rodea. Cualquier detalle en el universo de este tipo de juegos ayuda a que el jugador se sumerja en ellos. Este sistema pretende generar descripciones de elementos de estos juegos (estructuras, fauna, flora, bebidas, etc.), en este caso bebidas, para realzar su importancia y dotar de significado al transfondo del mundo.

A pesar de realizar un entrenamiento del modelo de lenguaje GPT-2 desde cero, y con un corpus reducido, los textos generados no son del todo malos. De las técnica de decodificación testeadas, es cierto que la que mejores resultados ofrece es la que tiene una menor perplejidad (*Beam Search (b=16)*), y no puede considerarse verdadero lenguaje natural. Aún así, el método *Nucleus (p=0.95)*, a pesar de presentar cierta incoherencia, tiene una perplejidad mayor, y las salidas que genera pretenden ser más creativas. Seguramente, el método *Nucleus (p=0.95)* presentaría mejores resultados entrenando el modelo de lenguaje con ajuste fino o con un corpus mayor.

En cuanto al futuro del TFG, se pretende ampliar el tamaño del corpus para ver su impacto en el texto generado por los métodos de codificación utilizados. Además, se pretende entrenar otro modelo de lenguaje, con el mismo corpus, pero con la técnica de ajuste fino, con la motivación de comparar los resultados de uno y otro. Otra idea que surgió durante el desarrollo del TFG, pero no se llevó a cabo, es la implementación de un caso de uso donde los usuarios pudieran interactuar con el modelo y evaluar los textos generados.

Bibliografía

Boden, M. A. (1998). Creativity and artificial intelligence. *Artificial Intelligence*, 103:347–356.

Carrigan, M., Debut, L., and Gugger, S. (2021). Transformer models - Hugging Face Course.

Colton, S., de Mántaras, R. L., and Stock, O. (2009). Computational creativity: Coming of age. *AI Magazine*, 30(3):11.

Cortez Vásquez, M. A., Vega Huerta, M. H., and Pariona Quispe, L. J. (2009). Procesamiento de lenguaje natural. *Revista de Ingeniería de Sistemas e Informática*, 6(2).

Dittmar, G. (2021). Natural Language Generation Part 2: GPT2 and Huggingface.

Education, I. C. (2021). Neural Networks.

Gkatzia, D., Lemon, O., and Rieser, V. (2017). Data-to-Text Generation Improves Decision-Making Under Uncertainty. *IEEE Computational Intelligence Magazine*, 12(3):10–17.

Guzdial, M., Harrison, B., Li, B., and Riedl, M. O. (2015). Crowdsourcing open interactive narrative.

Holtzman, A., Buys, J., Du, L., Forbes, M., and Choi, Y. (2020). The curious case of neural text degeneration.

Kapronczay, M. (2021). A beginner's guide to language models - Towards Data Science.

Pawade, M. D. and Sakhapara, M. A. (2018). Story scrambler - automatic text generation using word level rnn-lstm. *International Journal of Information Technology and Computer Science (IJITCS)*, 10(6):44–53.

Qiu, X., Sun, T., Xu, Y., Shao, Y., Dai, N., and Huang, X. (2020). Pre-trained models for natural language processing: A survey.

Sciforce (2019). A Comprehensive Guide to Natural Language Generation.

Vaca, A. (2021). Transformers en Procesamiento del Lenguaje Natural.

Vicente, M., Barros, C., and Lloret, E. (2018). Statistical language modelling for automatic story generation. *Journal of Intelligent & Fuzzy Systems*, 34(5):3069–3079.

Vicente, M., Barros, C., Peregrino, F. S., Agulló, F., and Lloret, E. (2015). La generación de lenguaje natural: análisis del estado actual. *Computación y Sistemas*, 19(4):721–756.

Wikipedia (2021). Perplexity.